НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«До захисту допущено»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *Кросбраузерний месенджер*

Виконав: студент 4 курсу, групи ІП-52

Циркун Владислав Вікторович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Головченко М.М.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *ст. викладач. к.т.н, Порєв В.М.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-ХХХХ. ХХХХХХ.02.81	<Тема ДП>.		
			Пояснювальна записка		
3	А4	КПІ.ІП-ХХХХ. ХХХХХХ.03.81	<Тема ДП>.		
			Технічне завдання		
4	А4	КПІ.ІП-ХХХХ. ХХХХХХ.04.51	<Тема ДП>.		
			Програма та		
			методика тестування		

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 17 таблиць та 10 джерел – загалом 60 сторінок.

Об’єкт дослідження: веб-застосунки з клієнт-серверною архітектурою у поєднанні з ORM для створення «віртуальної об’єктної бази даних», що застосовуються для роботи обміну миттєвими повідомленнями.

Мета дипломного проекту: створити кросбраузерну платформу для обміну миттєвими повідомленнями, що полегшує комунікацію між людьми on-line.

У першому розділі була описана та проаналізована предметна область, були проаналізовані успішні аналогічні проекти, а також був виконаний аналіз вимог до програмного забезпечення. Розроблено функціональні вимоги, схему варіантів використання та матрицю трасування.

У другому розділі була описана та розроблена архітектура застосунку. Побудовано структурні схеми бізнес процесів. Також був проведений аналіз безпеки даних.

У третьому розділі програмний продукт був протестований відповідно до розробленого тест планом. Були описані контрольні приклади тестів.

У четвертому розділі був послідовно описаний процес розгортання та впровадження веб-застосунку. Також були описані засоби супроводу.

КЛЮЧОВІ СЛОВА: МИТТЄВІ ПОВІДОМЛЕННЯ, КОМУНІКАЦІЙНА ПЛАТФОРМА, МЕСЕНДЖЕР, КРОСБРАУЗЕРНА ПЛАТФОРМА.

ABSTRACT

Explanatory note of the diploma project consists of 4 sections, contains 17 tables and 10 sources - total 60 pages.

Object of study: Client-server architecture web applications in conjunction with ORM to create a "virtual object database" that is used for instant messaging.

The aim of the diploma project: create cross-browser instant messaging platform that facilitates online communication between people.

The first section described and analyzed the subject area, analyzed successful similar projects, as well as an analysis of software requirements. Functional requirements, use case diagram and trace matrix are constructed.

In the second section, the architecture of the application was described and developed. Structure schemes of business processes are constructed. Data security analysis was also conducted.

In the third section, the software product was tested in accordance with the developed test plan. Control samples of the tests were described.

In the fourth section, the process of deployment and implementation of the web application was sequentially described. Means of support were also described.

KEYWORDS: MESSAGE COMMUNICATION, COMMUNICATION PLATFORM, MESSENGER, CROSS BROWSER PLATFORM.

ВМЕСТО ЭТОГО ЛИСТА ВСТАВИТЬ ЛИСТ ТИТУЛА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

PaaS – платформа як послуга.

API – набір визначень для взаємодії різнотипного програмного забезпечення.

BPMN – система умовних позначень (нотація) для моделювання бізнес-процесів.

MQTT – спрощений мережевий протокол, що працює на TCP / IP.

TCP – Протокол призначений для управління передачею даних у комп'ютерних мережах

UDP – Протокол, котрий виконує обмін повідомленнями без підтвердження та гарантії доставки.

ВСТУП

У наш час швидкого розвитку інформаційно-комунікаційних технологій можливість швидко обмінюватися інформацією необхідна як ніколи раніше. Досить часто потрібно постійно тримати зв'язок із родиною, замовником, друзями тощо. Також, на сьогоднішній день компаніям потрібно постійно комунікувати зі своїми працівниками, повідомляти про зміни, нововведення всередині компанії тощо.

Існує безліч сайтів та платформ із схожими цілями. Один з типів таких платформ є месенджер. Месенджери являються найкращими засобами для обміну інформації і переписки тому що не містять нічого зайвого, не використовують багато пам’яті і призначені для обмеженої кількості конкретних задач. Проте часто користувачі стикаються з не універсальністю, коли один месенджер стає некомфортними для виконання деяких задач і люди використовують декілька різних застосунків.

Метою цієї роботи є створення кросбраузерної платформи обміну миттєвими повідомленнями, що полегшує комунікацію між людьми on-line, яка використовуватимуть як у розважальних так і в комерційних цілях, а також котра об’єднує найбільш затребувані функціональні можливості вже існуючих месенджерів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Змістовний опис і аналіз предметної області

На сьогоднішній день у всесвітній павутині існує безліч сайтів та платформ, з різними цілями, однією із таких цілей є швидка комунікація між людьми по всьому світу. Саме для того, аби люди могли легко, а саме головне – швидко обмінюватись повідомленнями, а також для підтримки зв’язку з родичами та друзями створюються месенджери, соціальні мережі тощо. Кожна з цих систем має свою модель яка зручна для якогось із аспектів комунікації. Також такі платформи можуть мати свої особливості у функціоналі та додаткові можливості, що робить одну платформу кращу і популярнішу за іншу.

Звісно, різні сервіси обміну інформацією мають як відмінні так і спільні риси. В даному проекті буде створено месенджер, який об’єднує у собі різні, найбільш затребувані, особливості усіх аналогів, що власне і дозволить збільшити кількість потенційних користувачів.

1.2 Аналіз успішних IT-проектів

1.2.1 Аналіз відомих програмних продуктів

Серед схожих платформ можна виділити таких важливих конкурентів як Telegram, WhatsApp, Facebook Messenger, Viber, Instagram Direct і т д – їх досить багато, оскільки такий формат є зручним для переписки. Але між ними є різні відмінності, які полягають як у специфічних унікальних можливостях у функціоналі, так і в різних користувацьких інтерфейсах. Далі розглянуті найпопулярніші з них.

- Telegram. Cloud-based мобільний і настільний додаток для обміну повідомленнями, з акцентом на безпеку та швидкість. На мою думку – найзручніший месенджер на сьогоднішній день, який почав набувати своєї

популярності в Україні після заблокування соціальної мережі Вконтакте. Має простий, мінімалістичний інтерфейс, не містить зайвої інформації для користувача, зручний у користуванні. Новий користувач легко та швидко зможе розібратися в інтуїтивному інтерфейсі. Завоював популярність завдяки своїй швидкодії та захищеності. Для месенджера був створений протокол MTProto, що передбачає використання декількох протоколів шифрування. При авторизації і аутентифікації використовуються алгоритми RSA-2048, DH-2048 для шифрування, при передачі повідомлень протоколу в мережу вони шифруються AES з ключем, відомим клієнту і серверу. Також застосовуються криптографічні геш-алгоритми SHA-1 і MD5. [1]

- Facebook Messenger. Месенджер, який являється частиною соціальної мережі Facebook, яку використовує досить великий відсоток населення України для комунікації. Даний додаток побудований на базі відкритого протоколу MQTT. Має відносно зручний інтерфейс та багатший на додатковий функціонал, на відміну від Telegram, хоча більшість з нього не використовується і тільки ускладнює роботу у месенджері.

- Viber. Напевно один з найпопулярніших месенджерів в Україні, але в основному його використовує старше покоління і поступово його популярність зменшується. Настільна версія Viber використовує TCP і UDP порти 5242, 4244, 5243, 9785 і стандартні порти HTTP / HTTPS 80 і 443. Також протокол шифрування Viber використовує той же концепт, що і Протокол Signal. Якщо проводити його характеристику, то він схожий на Facebook Messenger, де також багато зайвого функціоналу.

Якщо підвести загальний підсумок, то на сьогоднішній день Telegram являється найшвидшим, найпростішим і найбільш захищеним месенджером на ринку. Але попри це користувачам інколи бракує деяких особливостей, які мають його конкуренти. У нашому продукті ми візьмем за основу простоту та

цікаві можливості Telegram та об’єднаємо з найпопулярнішими функціональними особливості інших месенджерів.

1.3 Аналіз вимог до програмного забезпечення

1.3.1 Розроблення функціональних вимог

Застосунок може мати безліч користувачів. В ході аналізу були виявлені такі необхідні користувачу варіанти використання:

- авторизація користувача;
- реєстрація користувача;
- створення групового чату;
- написання/відправка повідомлення;
- вхід користувача до групового чату;
- робота з чатом;
- робота з відправленими повідомленнями
- редагування особистих даних користувача;
- вихід з облікового запису.

Схема структурна варіантів використання, представлена у вигляді UML діаграми, наведена у Документі КПІ.ІП-5222.045440.06.99 СС Схема структурна варіантів використання застосунку.

Функціональні вимоги зазначені у таблиці 1.1.

Таблиця 1.1 - Функціональні вимоги

Варіант використання	Функціональна вимога	Пріоритет
Авторизація користувача	1. Застосунок має надавати можливість увійти в обліковий запис користувача	Високий

Продовження таблиці 1.1

Реєстрація користувача	2. Застосунок має надавати можливість користувачу реєструватися в системі	Високий
Створення групового чату	3. Застосунок має надавати можливість користувачу створювати групові чати	Високий
Написання/відправка повідомлення	4. Застосунок має надавати можливість користувачу відправляти повідомлення	Високий
Приєднання користувача до групового чату	5. Застосунок має надавати можливість користувачу шукати та приєднуватись до групових чатів	Високий
Робота з чатом	6. Застосунок має надавати можливість користувачу очищувати свої чати 6.1. Застосунок має надавати можливість користувачу очистити історію чату 6.2. Застосунок має надавати можливість користувачу видалити чат	Високий
Робота з відправленими повідомленнями	7. Застосунок має надавати можливість користувачу працювати з відправленими повідомленнями 7.1. Застосунок має надавати	Високий

Продовження таблиці 1.1

	можливість користувачу видаляти повідомлення 7.2. Застосунок має надавати можливість користувачу редагувати власні повідомлення	Високий
Редагування особистих даних користувача	8. Застосунок має надавати можливість користувачу редагувати особисті дані	Високий
Вихід з облікового запису	9. Застосунок має надавати можливість користувачу виходити із облікового запису	Високий

Для визначення залежності між варіантами використання та функціональними вимогами було розроблено матрицю трасування вимог, зображену на рисунку 1.1.

Варіант використання \ Функціональна вимога	1	2	3	4	5	6	6.1	6.2	7	7.1	7.2	8	9
Авторизація користувача	■												
Реєстрація користувача		■											
Створення групового чату			■										
Написання/відправка повідомлення				■									
Вхід користувача до групового чату					■								
Робота з чатом						■	■	■					
Робота з відправленими повідомленнями									■	■	■		
Редагування особистих даних користувача												■	
Вихід з облікового запису													■

Рисунок 1.1 – Схема структурна матриця трасування функціональних вимог

1.3.2 Розроблення нефункціональних вимог

Застосунок є кросбраузерним, тобто його можна буде запустити у будь-якому браузері на комп’ютері, смартфоні, планшеті та інших пристроях, які мають браузер. Для розробки обрано використати мову програмування JavaScript, з використанням фреймворку Vue 2 та VueX – патерн управління станами + бібліотека для застосунків на Vue.js. Також використано VueX-ORM для створення «віртуальної об’єктної бази даних». На сьогоднішній день Vue є

гнучким, швидким, реактивним фреймворком для створення веб застосунків. VueX ORM дозволяє створювати «нормалізовану» схему даних у VueX Store з такими відносинами, як «Has One» і «Belongs To Many».

Були виділені наступні нефункціональні вимоги:

- застосунок повинен працювати на пристроях з операційною системою Windows, Android, iOS, а також під ОС на базі ядра Linux та macOS;
- застосунок має працювати в будь-якому браузері;
- має бути передбачений захист від некоректних дій користувача;
- застосунок повинен мати простий та інтуїтивно зрозумілий інтерфейс.

1.3.3 Постановка комплексу завдань розробки

Призначенням розробки є ВЕБ-застосунок «Povidom Web» для обміну миттєвими повідомленнями у мережі Інтернет.

Метою розробки є полегшення людям процесу комунікації та зв’язку один із одним шляхом надання їм інтерфейсу через який вони можуть знайомитись та переписуватись on-line.

Для досягнення поставленої мети повинні бути вирішені наступні задачі:

- створення простого та інтуїтивного інтерфейсу користувача для задоволення функціональних вимог. Інтерфейс має бути зручним і легким у використанні;
- створення серверної частини застосунку для взаємодії інтерфейсу із сховищем даних;
- створення модулю роботи з даними. Модуль має надавати можливості збереження, вибірки та оновлення даних про користувача.

1.4 Висновки до розділу

В ході проведення аналізу вимог до розробки було детально розглянуто і проаналізовано предметне середовище – ринок месенджерів, оглянуті існуючі

технічні рішення та відомі програмні продукти (Telegram, Facebook Messenger, Viber). Виявлено що майже всі програмні продукти, описані вище, не є універсальними. Тому було вирішено що окремий застосунок, що матиме простий, інтуїтивний інтерфейс і матиме усі, справді необхідні, додаткові функціональні можливості на сьогоднішній буде актуальним.

Далі було досліджено сценарій користувача, розроблені функціональні та нефункціональні вимоги. Вибрано платформу для розробки ВЕБ-застосунку та технології (JavaScript, Vue 2, VueX, VueX-ORM, Node.js).

Сформульовані задачі розробки.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Перед створенням програмного продукту має бути проведений моделювання та аналіз програмного забезпечення. Для цього ми будем використовувати діаграми BPMN.

Нижче описані основні процеси, використання застосунку користувачем.

а) Авторизація користувача у месенджері.

1) Користувач заходить на сторінку входу.

2) Вводить данні для авторизації.

3) Дані відправляються на сервер.

4) Якщо дані коректні, то створюється токен та сесія для користувача і його перенаправляє на головну сторінку застосунку.

5) Якщо введені дані – некоректні, то висвічується повідомлення про помилку.

б) Створення групового чату.

1) У головному меню користувач обирає опцію «New Group» і відкривається модальне вікно створення групового чату.

2) Вводить данні нового групового чату.

3) Відправляється запит на сервер на створення.

4) Якщо помилок не виявлено, то сервер повідомляє користувача про успішне проходження запиту.

5) Інакше повідомляє користувача про те, що вже є груповий чат із таким ім’ям.

в) Оновлення налаштувань акаунту.

1) У головному меню користувач обирає опцію «Settings» і відкривається модальне вікно із налаштуванням акаунту.

2) Користувач редагує налаштування.

3) Данні відправляються на сервер.

4) У випадку коли дані коректні, то все оновлюється і повертаються користувачу.

5) Інакше висвічується повідомлення про помилку оновлення.

г) Реєстрація користувача.

1) Відкривається сторінка реєстрації.

2) Користувач заповнює усі необхідні поля для реєстрації.

3) Відправка запиту на реєстрацію на сервер.

4) Якщо дані не унікальні, то користувач повідомляється про некоректність даних.

5) Якщо дані не пройшли умови безпеки, то користувач повідомляється про невідповідність параметрам безпеки.

6) Якщо введені дані пройшли всі перевірки успішно, то створюється обліковий запис і користувач повідомляється про це.

Основні бізнес-процеси застосунку відповідають основним принципам роботи застосунку, наведеним у попередньому підрозділі, та включають авторизацію користувача у месенджері, створення групових чатів, оновлення налаштувань акаунту та реєстрацію користувача.

Схеми структурні бізнес процесів, представлені у вигляді BPMN нотації, наведені у Документі КПІ.ІП-5222.045440.06.99 СС Схема структурна бізнес-процесів застосунку.

2.2 Архітектура програмного забезпечення

Розроблена система виконується під операційною системою Windows. Використання під операційними системами на базі ядра Linux та macOS можливе, але не було протестоване.

Для розробки браузерної частини системи було застосовано мову JavaScript, з використанням фреймворку Vue 2. Використано патерн управління станами VueX, який слугує централізованим сховищем даних для всіх компонентів програми з правилами, що гарантують, що стан може бути змінено тільки передбачуваним чином. Сховище VueX являється реактивне, тому коли компоненти Vue покладаються на його стан, то вони будуть реактивно і ефективно оновлюватися, якщо стан сховища змінюється.

Для розробки серверної частини було застосовано Node.js у поєднанні з VueX-ORM для створення «віртуальної об’єктної бази даних». VueX-ORM являє з себе плагін для VueX, який забезпечує доступ об'єктно-реляційного зіставлення до VueX Store. VueX ORM дозволяє створювати «нормалізовану» схему даних у VueX Store з такими відносинами, як «Has One» і «Belongs To Many». Він також надає вільний API для отримання, пошуку та оновлення стану у Store.

Загальна структура ВЕБ-додатку представлена на Рис 2.1

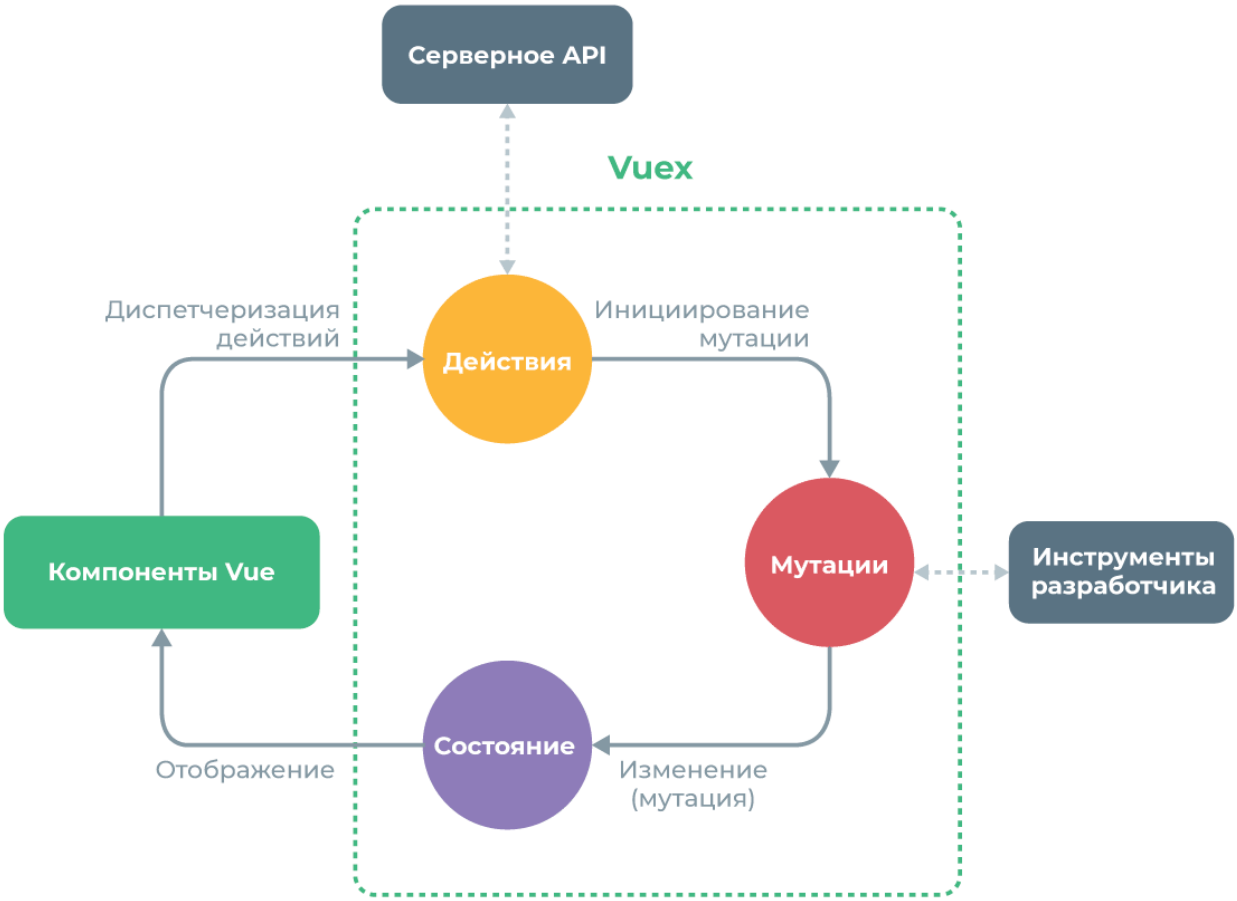

Рисунок 2.1 – Загальна структура ВЕБ-додатку

Клієнтська частина складається з точки входу – яка в залежності від шляху запиту відображає потрібний компонент. Можна розділити компоненти на 2 типи, компоненти - основних інтерфейсних блоків програми та міні-компоненти, які багатократно використовуються в різних компонентах, вони представляють з себе окремі елементи інтерфейсу.

2.3 Конструювання програмного забезпечення

2.3.1 Опис архітектури програмного забезпечення

Опишемо конструктивні складові серверної частини застосунку. Опис спроектованих модулів та класів наведено у таблиці 2.1.

Таблиця 2.1 – Опис основних класів додатку

Модуль	Клас	Призначення

Продовження таблиці 2.1

Controllers	AppController.AuthController	Клас, що містить логіку обробки авторизаційних даних на головному інтерфейсі застосунку
	AuthController.AuthController	Клас, що містить логіку обробки авторизаційних даних при реєстрації та логіюванні та системну логіку роботи з авторизацією
	ChatController.ChatController	Клас, що містить логіку роботи з чатами різних типів
	MessageController.MessageController	Клас, що містить логіку обробки повідомлень
	UserController.UserController	Клас, що містить логіку роботи з користувачами
Middleware	AuthCheck.AuthCheck	Клас, що містить логіку перевірки паролів, токенів при авторизації

Продовження таблиці 2.1

Events	Index.Events	Клас, що відповідає за опис та обробку подій

2.3.2 Архітектурні компоненти програмного забезпечення

Модуль роботи з базою даних. Так як використовується VueX-ORM, то створено декілька моделей, які описують структуру збереження даних. Їх опис наведений у таблицях 2.2 – 2.5.

Таблиця 2.2 - Опис моделі ChatModel

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор
type	String	Визначається тип чату, або груповий або приватний
title	String	Заголовок діалогу
avatar	String	Шлях до зображення у пам’яті
active	Boolean	Помітка чи співрозмовник у мережі
members	Array	Список

Продовження таблиці 2.2

		учасників чату
messages	Object	Приймає об’єкт, який повертає модель Message, має властивість hasMany і зв’язується з по зовнішньому ключу chat_id

Таблиця 2.3 - Опис моделі DialogModel

Поле	Тип	Опис
notifications	Boolean	Звукові сповіщення
unreadCount	Number	Кількість непрочитаних повідомлень
messages	ref: message	Посилається на MessageModel. Список усіх повідомлень
chat	ref: chat	Посилається на ChatModel

Продовження таблиці 2.3

companion	ref: user	Посилається на UserModel

Таблиця 2.4 - Опис моделі MessageModel

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор
chat_id	ObjectID	Унікальний ідентифікатор, який здійснює прив'язку повідомлення до конкретного чату
content	String	Вміст повідомлення
author_id	ObjectID	Унікальний ідентифікатор, який здійснює прив'язку повідомлення до конкретного користувача
edited	Boolean	Помітка що повідомлення

Продовження таблиці 2.4

		відредаговане
read	Boolean	Помітка що повідомлення прочитане

Таблиця 2.5 - Опис моделі UserModel

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор
username	String	Унікальний юзернейм користувача
password	String	Пароль користувача
online	Boolean	Відмітка чи користувач у мережі
lastActivity	Number	Час, коли в останє був у мережі
avatar	String	Шлях до зображення(аватарки користувача) у

Продовження таблиці 2.5

		пам’яті
palette	Number	Фон аватарки
dialogs	Number	Посилається на DialogModel

Опис основних методів по всім класам та моделям наведено нижче у таблиці 2.6 .

Таблиця 2.6 - Опис основних методів класів та моделей

Клас/Модель	Метод	Опис
UserModel	hashPassword(password)	Метод, який описує реалізацію хешування пароля за допомогою хеш функції bcrypt
	validPassword(password)	Перевірка пароля на валідність
	generateJWT()	Метод генерування нового унікального токена для користувача на момент створення нової сесії
	updateStatus(userID, {online, lastActivity})	Метод, який оновлює статус користувача (online/offline), в залежності від того чи він знаходиться у застосунку

Продовження таблиці 2.6

AuthCheck	socket(socket, next)	Метод, який використовуючи проміси присвоює користувачу новий токен, а якщо токен ще не було створено, то відправляється reject з текстом помилки і цей момент відслідковується і перехвачується
UserController	initRoutes	Метод, у якому ініціалізуються роути пошуку
	search(req, res)	Метод, який отримуючи значення поля пошуку користувачів - зіставляє його із юзернеймами зареєстрованих користувачів
MessageController	create(req, res)	Метод, який створює повідомлення і записує інформацію про повідомлення у MessageModel
	update(req, res)	Метод, який дозволяє редагувати відправлене повідомлення і потім

Продовження таблиці 2.6

MessageController		перезаписує відповідну інформацію про повідомлення в MessageModel
	delete(req, res)	Метод, який дозволяє видалити відправлене повідомлення, як і тільки для себе так і для співбесідника і потім видаляє інформацію про повідомлення в MessageModel
	readMessage(req, res)	Метод, який змінює статус повідомлення на прочитане в MessageModel
ChatController	deleteDialog(req, res)	Метод, який очищає історію діалогу, а потім видаляє його з Store та видаляє відповідний об’єкт з ChatModel
	setNotifications(req, res)	Метод, який дозволяє встановлювати звукові сповіщення на конкретному діалозі, та заносить відповідну інформацію у

Продовження таблиці 2.6

ChatController		DialogModels
	clearHistory(req, res)	Метод, який очищає історію діалогу у Store та очищує інформацію полів відповідного об’єкту у ChatModel
	getDialog(req, res)	Метод, який здійснює пошук діалогів користувача, у яких він перебуває, порівнюючи введене значення у полі пошуку, та значення заголовків чатів у об’єктах ChatModel
	createGroup(req, res)	Метод, який створює новий обйєкт типу ChatModel для створення групового чату, і заповнює нову інформацію у поля нового об’єкту
AuthController	login(req, res)	Метод, який описує процес залогінення користувача в обліковий запис, а саме перевіряє на валідність пароль, перевіряє чи існує такий

Продовження таблиці 2.6

AuthController		користувач у базі та у разі помилок видає помилки. Також, у разі успішного проходження всіх умов у цьому методі викликається функція створення токену сесії
	register(req, res)	Метод, який описує процес створення облікового запису користувача у системі. Перевіряється чи не існує вже користувач із обраним юзернеймом, а також у разі успішного проходження всіх умов викликається функція хешування пароля і зі всіма даними зберігається в об’єкті моделі UserModel. А також, викликається функція створення токену сесії

Далі розглянемо структуру клієнтської частини застосунку. Вона складається з модулів логіки, віджетів та компонентів.

Щоб підвищити ефективність, природно шукати правильні способи поділу роботи на дільниці з мінімальними перетинами між людьми і підсистемами.

Впровадження компонентного підходу (в цілому) - це те, як зазвичай вирішується таке завдання. Будь-яка компонентна система повинна зменшувати загальну складність через надання ізоляції, або природних бар'єрів, що приховують складність одних систем від інших. Хороша ізоляція також полегшує повторне використання та впровадження сервісних парадигм.

Логіка клієнтських веб-додатків в таких ситуаціях істотно ускладняється, іноді вона навіть стає складніше, ніж на серверній стороні. Можливим вирішенням даної складності може бути подальший поділ на компоненти і ізоляція логіки всередині однієї сторінки або документа. Саме тому було вирішено використовувати компонентний підхід побудови застосунку.

Логіка клієнтської частини містить код для обробки даних та для взаємодії з серверною частиною застосунку.

Віджети – це глобальні інтерфейсні елементи застосунку.

Компоненти – це допоміжні інтерфейсні елементи застосунку, які створюються для багато повторного використання. Усі вище названі елементи наведені у таблиці 2.7.

Таблиця 2.7 - Перелік конструктивних елементів клієнтської частини застосунку

Назва	Тип	Опис
Authorize	Віджет	Даний віджет - це інтерфейс сторінки авторизації з 2 вкладками: Sign in та Sign up

Продовження таблиці 2.7

App	Віджет	Даний віджет – це інтерфейс головної сторінки застосунку, на яку переходить користувач після залогінення/реєстрації
NavigationView	Компонент	Компонент, який відображає меню навігації, яке вилітає після натискання кнопки меню(бутерброд)
VAlert	Компонент	Компонент, який відображає спливаючий блок з попередженням/підказкою
Multipane, MultipaneResizer	Компонент	Компоненти, які забезпечують зміну ширини списку діалогів відносно головного вікна чату та містять логіку обрахування ширин цих блоків
ChatList	Компонент	Компонент, який відображає ліву

Продовження таблиці 2.7

		частину застосунку: список діалогів, поле пошуку, кнопка меню (бутерброд)
ChatView	Компонент	Компонент, який відображає праву частину застосунку, а саме переписку, поле вводу повідомлення та та шапку з інструментами та всією інформаціює про діалог
Users	Компонент	Компонент для пошуку користувачів застосунку, який представляє з себе модальне вікно
Settings	Компонент	Компонент, який відображає налаштування акаунту і має вигляд модального вікна
NewGroup	Компонент	Компонент для створення нового групоого чату, який представляє з себе

Продовження таблиці 2.7

		модальне вікно
ChatToolbar	Компонент	Компонент, який відображає шапку з інструментами у головному вікні чату
ChatMessages	Компонент	Компонент, який відображає вікно зі списком усіх повідомлент , які були надіслані в діалозі
ChatTextarea	Компонент	Компонент, який відображає поле вводу нового повідомлення у поточний діалог, або поле для редагування старого повідомлення
ChatNewMessageArea	Компонент	Компонент, який відображає поле вводу нового повідомлення у поточний діалог
ChatEditMessageArea	Компонент	Компонент, який відображає поле для редагування відправленого раніше повідомлення

Продовження таблиці 2.7

ChatDateLabel	Компонент	Компонент, який відобража ярлик із днем відправки усіх повідомлень, які ідуть після нього
ChatMessage	Компонент	Компонент, у якому відображається текст повідомлення, час відправки, іконка, яка свідчить про те, чи прочитав/не прочитав його співбесідник, а також ярлик з написом «edited», якщо повідомлення було відредактовано
Avatar	Компонент	Компонент, у якому відображаєься кружечок з аватаркою користувача, або просто кружечок з рандомно згенерованим фоном і літерами ініціалів, якщо фотографії нема
ChatUnreadLabel	Компонент	Компонент, який відображає ярлик, з

Продовження таблиці 2.7

		текстом «Unread messages», у випадку, якщо у поточному діалозі є непрочитані повідомлення
ChatDeleteMessage	Компонент	Компонент, який відображає модальне вікно, у якому є можливість обрати варіант видалити повідомлення як для себе, так і для співбесідника, яке з’являється, коли користувач хоче видалити власне повідомлення
ChatListItem	Компонент	Компонент, який відображає діалог (груповий або приватний) у списку діалогів, які потім перебирають у циклі, щоб вивести весь список існуючих діалогів користувача

Продовження таблиці 2.7

Scroller	Компонент	Компонент, який відображає полосу завантаження
Wrapper	Компонент	Компонент, який являється глобальною обгорткою для більших компонентів
CropAvatar	Компонент	Компонент, який відображає модальне вікно, у якому є можливість обрати та завантажити аватарку профілю. А міститься логіка обрізання фото

Розглянемо опис основних функцій по класам клієнтської частини в таблиці 2.8, що буде наведена нижче.

Таблиця 2.8 - Перелік основних функцій по класам клієнтської частини

Клас	Метод	Опис дії
store.Auth	login(credentials)	Метод, що приймає у Store-і значення заповнених полів під час входу в акаунт, і перенаправляє в povidom.auth, передаючи об’єкт

Продовження таблиці 2.8

povidom.auth	login(data)	Метод, що приймає дані логінування і відправляє post запит на сервер
store.Auth	register(credentials)	Метод, що приймає у Store-і значення заповнених полів під час реєстрації нового акаунту, і перенаправляє в povidom.auth, передаючи об’єкт
povidom.auth	register(data)	Метод, що приймає реєстраційні дані нового користувача і відправляє post запит на сервер
store.Auth	logout({commit})	Метод, у якому state авторизації задають як false, перенаправляють в povidom.auth і потім змінюють поточний роут на роут сторінки авторизації
povidom.auth	logout()	Метод, у якому видаляють токен сесії

Продовження таблиці 2.8

povidom.chats	getDialog(params)	Метод, який відправляє get запит на сервер для отримання діалогу
povidom.chats	createGroup(data)	Метод, який відправляє post запит на сервер для створення нового групового чату
povidom.chats	clearHistory(chatID)	Метод, який відправляє get запит на сервер із конкретним chatID для очищенні історії чату
povidom.chats	deleteDialog(chatID)	Метод, який відправляє get запит на сервер із конкретним chatID для видалення чату
povidom.chats	setNotifications(data)	Метод, який відправляє get запит на сервер щоб активувати/вимкнути сповіщення
povidom.messages	getMessages(params)	Метод, що відправляє

Продовження таблиці 2.8

		get запит на сервер, щоб отримати масив повідомлень
povidom.messages	sendMessage(message)	Метод, який відправляє post запит на сервер, щоб зберегти відправлене повідомлення співбесіднику
povidom.messages	updateMessage(message)	Метод, який відправляє put запит на сервер, щоб записати відредаговані дані повідомлення в об’єкт
povidom.messages	deleteMessage(message)	Метод, який відправляє delete запит на сервер, щоб видалити обраний об’єкт повідомлення зі сховища
povidom.token	remove()	Метод, який видаляє токен активної сесії користувача
povidom.token	isValid()	Метод, який перевіряє токн на валідність і

Продовження таблиці 2.8

		повертає Boolean значення
povidom.token	getExpirationDate(encodedToken)	Метод, який повертає дату закінчення терміну дії токену активної сесії
povidom.token	isExpired (encodedToken)	Метод, який перевіряє чи закінчився термін дії токену сесії і повертає Boolean значення
povidom.user	setAvatar(data)	Метод, який відправляє post запит на сервер, щоб зберегти шлях до нової аватарки користувача у модель UserModel
povidom.user	search (params)	Метод, який відправляє get запит на сервер, щоб отримати відповідь по пошуку користувачів у системі

2.4 Аналіз безпеки даних

Щоб дані, які зберігаються у БД були захищені, я використовую функцію хешування bcrypt + сіль та зберігаю паролі користувачів захешованими. Такий метод вважається досить сильним для забезпечення надійності застосунку, оскільки це хешування має досить великий час роботи, що не дає його використовувати при підстановках паролю. Також використовуються унікальні токени на період кожної сесії користувача у системі.

2.5 Висновки по розділу

Застосунок складається з клієнтської та серверної частин, що відповідають за необхідні частини роботи застосунку.

Для розробки клієнтської частини було вирішено використовувати компонентний підхід побудови застосунків тому що така система повинна зменшувати загальну складність через надання ізоляції, або природних бар'єрів, що приховують складність одних систем від інших.

Клієнтська частина розроблялась із використанням фреймворку Vue, що дозволило зробити застосунок швидким та забезпечує реактивність.

Замість створення бази даних я вирішив використати VueX-ORM який являє з себе плагін для VueX, який забезпечує доступ об'єктно-реляційного зіставлення до VueX Store. VueX ORM дозволяє створювати «нормалізовану» схему даних у VueX Store.

Застосунок працює у режимі on-line, дані користувача зберігаються у хмарі. Під час розробки використовувалась система контролю версій git.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Етап тестування програмного забезпечення відіграє важливу роль у циклі розробки. Також, якість програмного забезпечення є важливою для клієнт-серверних багатокористувацьких сервісів.

Тестування ВЕБ-застосунків – складний процес тому що треба враховувати різні розміри екрану, апаратні відмінності, різні типи браузерів та їх версій, різні типи підключення до інтернету, раптові обриви зв'язку тощо. Для тестування QA спеціалісти використовують різні інструменти за засоби тестування.

Після того як послідовно були проведені різні тести застосунку можна підтвердити правильність, функціональну поведінку та зручність використання, перш ніж публікувати його.

Тестування надає такі переваги:

- стабільна швидкість розвитку, що допомагає мінімізувати технічні втрати;
- завчасне виявлення помилок у циклі розробки;
- швидкий відгук про помилки;
- рефакторинг коду, що дозволяє оптимізувати його.

Тестування веб-застосунків має свої особливості:

- для тестування потрібні різні браузери та різні їх версії;
- у екранів пристроїв різні розміри;
- широкий спектр конкретних операційних систем та компонентних конфігурацій;
- на сьогоднішній день версії браузерів швидко застарівають.

У випадку якщо користувач захоче зайти у застосунок з мобільного пристрою, то тут виникають інші особливості тестування:

- мобільні пристрої використовують різні мережеві підключення (3G, 4G, Wi-Fi), що менш стабільні за широкосмугове для ПК;
- треба тестувати застосунок з різною швидкістю передачі даних тому що мобільні пристрої постійно здійснюють пошук мережі.

До плану тестування ввійде набір функцій, які будуть протестовані. План описує програмні об’єкти що будуть протестовані, тип тестів, необхідні ресурси та план необхідний для виконання тестування.

Дотримання найкращих практик тестування веб-застосунків та ретельна перевірка всієї функціональності застосунку є дуже важливим для його успіху.

У даному плані будуть протестовані наступні функції:

- авторизація користувачів;
- реєстрація користувачів;
- створення групових чатів;
- редагування налаштувань акаунту;
- пошук користувачів;
- відправка повідомленнь;
- редагування та видалення повідомлень.

Налаштовані наступні тестові модулі:

- авторизація користувача;
- авторизація користувача з невалідним паролем;
- реєстрація користувача;
- реєстрація користувача з невалідними полями;
- створення групових чатів;
- редагування та видалення повідомлень;
- пошук користувачів;
- відправка повідомлень;
- редагування налаштувань акаунту.

3.2 Підходи до тестування

Підходи до тестування наведені в документі «Програма та методика тестування» дипломного проекту (КПІ.ІП-5222.045440.04.51).

3.3 Критерії проходження тестування

Критерії проходження тестування наведені в документі «Програма та методика тестування» дипломного проекту (КПІ.ІП-5222.045440.04.51).

3.4 Процес тестування

3.4.1 Дані до тестів

Вхідними даними тесту швидкодії є набори різноманітних даних які імітують натуральне використання програмного забезпечення і покривають усі варіанти роботи системи у конкретному випадку. Вихідними даними являються кількість оброблених запитів, швидкості обробки запитів, дані по навантаженню (завантаження мережі, навантаження процесору тощо).

Вхідними даними інтеграційного тесту є набори повідомлень, який отримує компонент системи ззовні від інших компонентів відповідно до конкретного тесту. Вихідними даними є результати роботи компоненту та його підлеглих.

Вхідними даними компонентного тесту є набори параметрів, контекстів та визначених результатів, які і є вихідними даними компонентного тесту.

3.4.2 Задачі тесту

Кожен тест повинен перевірити як правильність програми у відповідності до умов виконання тесту, так і виявити можливі помилки у роботі.

3.5 Вимоги до середовища

Вимоги до середовища наведені в документі «Програма та методика тестування» дипломного проекту (КПІ.ІП-5222.045440.04.51).

3.6 Опис контрольного прикладу

Нижче описане тестування здійснене для Веб-застосунку для обміну миттєвими повідомленнями, що полегшує комунікацію між людьми on-line, написаного зі мові програмування JavaScript.

Застосунок проходив тестування на HP Envy dv7 на ОС Windows 10. Було протестовано на всіх можливих розширеннях екранів завдяки панелі інструментів веб розробника chrome dev tools. А також були використані засоби сайту BrowserStack для тестування застосунку на різних пристроях з різноманітним розширенням екранів, різних операційних системах та на різних браузерах.

Таблиця 3.1 – Перше використання

Мета тесту	Перевірка поведінки застосунку при першому використанні.
Початковий стан	Застосунок встановлено. Користувач ще не використовував застосунок.
Вхідні дані	-
Схема проведення тесту	Відкрити застосунок у браузері за посиланням.
Очікуваний результат	Відкривається сторінка авторизації, на якій знаходиться 2 вкладки: «Sign in» та «Sign up» та кнопка «Next» і користувач може обирати або увійти в існуючий обліковий запис, а якщо ще не створив його, то перейти у вкладку «Sign up»

Продовження таблиці 3.1

	та створити його.
Фактичний результат	Відкривається сторінка авторизації, на якій знаходиться 2 вкладки: «Sign in» та «Sign up» та кнопка «Next» і користувач може обирати або увійти в існуючий обліковий запис, а якщо ще не створив його, то перейти у вкладку «Sign up» та створити його.

Таблиця 3.2 – Авторизація користувача з невідповідними даними

Мета тесту	Перевірка поведінки застосунку при введенні даних про неіснуючих користувачів у поля авторизації.
Початковий стан	Користувач знаходиться на сторінці авторизації у вкладці «Sign in».
Вхідні дані	Значення полів Username та Password.
Схема проведення тесту	Ввести у поле Password пароль, який не відповідає юзернейму у полі Username або навпаки і натиснути на кнопку Next.
Очікуваний результат	З’являється спливаюче вікно попередження, у якому написано «Wrong credentials».
Фактичний результат	З’являється спливаюче вікно попередження, у якому написано «Wrong credentials».

Таблиця 3.3 – Авторизація користувача, за умови, що хоча б одне поле лишається не заповнене

Мета тесту	Перевірка поведінки застосунку при спробі авторизуватись у системі лишивши хоча б одне поле форми авторизації – пустим.
Початковий стан	Користувач знаходиться на сторінці авторизації у вкладці «Sign in».
Вхідні дані	Значення полів Username та Password.
Схема проведення тесту	Заповнити тільки 1 поле або лишити 2 поля пустими.
Очікуваний результат	Кнопка Next буде неактивна та заблокована, а також коли переходиш з одного поля на інше, лишивши його пустим з’являється підказка із червоним написом «This field is required». Також рамочка та підказка до незаповненого поля стають червоними.
Фактичний результат	Кнопка Next буде неактивна та заблокована, а також коли переходиш з одного поля на інше, лишивши його пустим з’являється підказка із червоним написом «This field is required». Також рамочка та підказка до незаповненого поля стають червоними.

Таблиця 3.4 – Реєстрація користувача, за умови, що поля із паролем і його підтвердженням не збігаються

Мета тесту	Перевірка поведінки застосунку при неправильному повторному вводі паролю.
Початковий стан	Користувач знаходиться на сторінці авторизації у вкладці «Sign up».
Вхідні дані	Значення полів Username, Password, Confirm Password.
Схема проведення тесту	Ввести значення у поля Username та Password, а у поле Confirm Password ввести щось, що не співпадає із полем Password.
Очікуваний результат	Кнопка Next буде неактивна та заблокована, а також під полем Confirm Password з’являється підказка із написом червоним кольором «Passwords don't match». Також рамочка та підказка до даного поля стають червоними.
Фактичний результат	Кнопка Next буде неактивна та заблокована, а також під полем Confirm Password з’являється підказка із написом червоним кольором «Passwords don't match». Також рамочка та підказка до даного поля стають червоними.

Таблиця 3.5 – Реєстрація користувача, за умови, що хоча б одне поле лишається не заповнене

Мета тесту	Перевірка поведінки застосунку при спробі

Продовження таблиці 3.5

	зареєструватися у системі лишивши хоча б одне поле форми реєстрації – пустим.
Мета тесту	Перевірка поведінки застосунку при спробі зареєструватися у системі лишивши хоча б одне поле форми реєстрації – пустим.
Початковий стан	Користувач знаходиться на сторінці авторизації у вкладці «Sign up».
Вхідні дані	Значення полів Username, Password, Confirm Password.
Схема проведення тесту	Заповнити тільки 1 або 2 поля. Або лишити всі поля пустими.
Очікуваний результат	Кнопка Next буде неактивна та заблокована, а також коли переходиш з одного поля на інше, лишивши його пустим з’являється підказка із червоним написом «This field is required». Також рамочка та підказка до незаповненого поля стають червоними.
Фактичний результат	Кнопка Next буде неактивна та заблокована, а також коли переходиш з одного поля на інше, лишивши його пустим з’являється підказка із червоним написом «This field is required». Також рамочка та підказка до незаповненого поля стають червоними.

Таблиця 3.6 – Реєстрація користувача, за умови, що користувач із запропонованим Username вже існує

Мета тесту	Перевірка поведінки застосунку при вже

Продовження таблиці 3.6

	існуючому у базі Username, який запропонував новий незареєстрований користувач.
Початковий стан	Користувач знаходиться на сторінці авторизації у вкладці «Sign up».
Вхідні дані	Значення полів Username, Password, Confirm Password.
Схема проведення тесту	Ввести значення у поля Username, Password та Confirm Password та натиснути кнопку Next. Значення поля Username має містити нік юзера, який вже зареєстрований.
Очікуваний результат	З’являється спливаюче вікно попередження, у якому написано «The user with such username already exists».
Фактичний результат	З’являється спливаюче вікно попередження, у якому написано «The user with such username already exists».

Таблиця 3.7 – Пошук користувачів у месенджері

Мета тесту	Перевірка поведінки застосунку при виконанні пошуку інших користувач у застосунку.
Початковий стан	Користувач увійшов в обліковий запис.
Вхідні дані	Значення поля пошуку по Username.
Схема проведення тесту	Натиснути на значок меню у верхньому лівому

Продовження таблиці 3.7

	кутку, після висування меню, зі списку обрати Users та у полі пошуку почати вводити Username користувача, якого хочеш знайти.
Очікуваний результат	При введені значення у поле пошуку починається з’являтися список користувачів, username яких має літери у такому ж порядку, як було введено у поля пошуку.
Фактичний результат	При введені значення у поле пошуку починається з’являтися список користувачів, username яких має літери у такому ж порядку, як було введено у поля пошуку.

Таблиця 3.8 – Пошук неіснуючих користувачів у месенджері

Мета тесту	Перевірка поведінки застосунку при виконанні пошуку неіснуючих користувач у застосунку.
Початковий стан	Користувач увійшов в обліковий запис.
Вхідні дані	Значення поля пошуку по Username.
Схема проведення тесту	Натиснути на значок меню у верхньому лівому кутку, після висування меню, зі списку обрати Users та у полі пошуку почати вводити Username користувача, якого хочеш знайти.
Очікуваний результат	При введені неіснуючого Username-у у системі, з’явиться підказка, на якій буде написано «No users were found».

Продовження таблиці 3.8

Фактичний результат	При введені неіснуючого Username-у у системі, з’явиться підказка, на якій буде написано «No users were found».

3.7 Висновки до розділу

Тестування веб-застосунків є досить цікавим та має багато своїх особливостей, а саме різні браузери та їх різні версії, різні розміри екранів пристроїв, широкий спектр конкретних операційних систем та компонентних конфігурацій, апаратні відмінності, версії браузерів швидко застарівають, різні типи підключення до інтернету, раптові обриви зв'язку тощо.

Попри це, тестування під час розробки надає багато переваг. Серед них можна виділити наступні:

- стабільна швидкість розвитку, що допомагає мінімізувати технічні втрати;
- завчасне виявлення помилок у циклі розробки;
- швидкий відгук про помилки;
- рефакторинг коду, що дозволяє оптимізувати його;
- стабільна швидкість розробки, що веде до зменшення витрат.

Після того як послідовно були проведені різні тести застосунку можна підтвердити правильність, функціональну поведінку та зручність використання, перш ніж публікувати його.

У розділі наведені тестування зручності використання, сумісності, виконання, тести для пошуку користувачів, збережених у Store, створеного з використанням мови JavaScript та фреймворку Vue та VueX-ORM для створення «віртуальної об’єктної бази даних» яка представляє з себе «нормалізовану» схему даних у VueX Store.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для повного розгортання ВЕБ-застосунку потрібен комп’ютер із встановленою операційною системою Windows (або, під ОС на базі ядра Linux, macOS), а також будь-який веб-браузер.

4.1.1 Встановлення основного сервісу

Потрібно взяти файли директорії /povidom-vue та помістити в необхідну робочу папку.

4.1.2 Встановлення Node.js

Необхідно перейти за посиланням «https://nodejs.org/en» та встановити останню версію Node.js, а разом з ним автоматично встановиться і npm модуль.

4.1.3 Запуск основного сервісу

Щоб запустити сервер у командному рядку cmd.exe з папки /povidom-vue необхідно виконати команду npm install, а потім npm run serve.

4.1.4 Розгортання в on-line режимі

Після публікації на Heroku(https://www.heroku.com) застосунок буде доступним для використання звідти. В такому разі користувачу буде потрібно знайти застосунок за назвою «Povidom Web».

4.2 Робота з програмним забезпеченням

Інструкцію роботи із застосунком наведено в документі «Керівництво користувача» дипломного проекту (КПІ.ІП-5222.045440.05.34).

4.3 Супровід програмного забезпечення

Після того як застосунок був розміщений на платформі Heroku - хмарна PaaS-платформа, що підтримує ряд мов програмування [2], для контролю виникнення проблем, які не було відтворено на етапах розробки і тестування, а також для додаткового тестування продукту використовуються відповідні засоби платформи.

4.4 Висновки до розділу

У розділі було описано процес розгортання програмного забезпечення, було описано про варіант розгортання в on-line режимі, наведено інструкцію користувача, а також зазначено про можливі варіанти та засоби супроводу застосунку.

ВИСНОВКИ

У ході розробки дипломного проекту було здійснено аналіз предметного середовища. Були проаналізовані існуючі веб-застосунки та розглянуті різноманітні підходи та технічні рішення для вирішення даної задачі. Зрештою, вдалось виявити основні глобальні недоліки конкурентів і в результаті було сформовано вимоги для кінцевого застосунку.

Спроектований і розроблений веб-застосунок для пристроїв з будь-яким браузером полегшує процес спілкування та обміну миттєвими повідомленнями між людьми on-line. Його можна буде використовувати як у розважальних так і в комерційних цілях.

Варто підкреслити наступні вагомі перевагами даного застосунку перед іншими аналогами:

- простий та інтуїтивний дизайн;
- універсальність – застосунок, що об’єднує найбільш затребувані та справді необхідні додаткові функціональні можливості вже існуючих аналогів;
- висока продуктивність, яка досягнута за рахунок використання нових технологій та сучасних підходів вирішення популярних проблем.

У розділі «Моделювання та конструювання програмного забезпечення» здійснив моделювання застосунку та розробив архітектуру. Було вирішено використовувати компонентний підхід у розробці клієнтської частини щоб зменшити загальну складність через надання ізоляції. Хороша ізоляція також полегшує повторне використання та впровадження сервісних парадигм. Після чого було проведене конструювання.

Після завершення основного етапу розробки було проведено тестування. Було обґрунтовано необхідність тестування застосунку, надано декілька контрольних прикладів тестів та усунені знайдені недоліки.

У розділі «Впровадження ту супровід програмного забезпечення» були надані детальні інструкції по впровадженню застосунку, описані засоби супроводу. Також було розроблене керівництво користувача.

Були спроектовані різні графічні матеріали: схеми варіантів використання, схеми бізнес-процесів застосунку.

Підбиваючи загальні підсумки, можна сказати, що розроблений веб-застосунок являється повноцінним програмним продуктом, який може активно використовуватися людьми для спілкування та застосовуватися у різних сферах діяльності. Розроблений месенджер надійно протестований і у перспективі є проведення більш глибокого аналізу цільової аудиторії задля удосконалення застосунку, покращення системи захисту персональних даних, швидкодії і додання нових можливостей, яких не вистачає користувачам.

ПЕРЕЛІК ПОСИЛАНЬ

1) Telegram [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Telegram
2) Heroku [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Heroku
3) Інтеграційне тестування [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Інтеграційне_тестування
4) Модульне тестування [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Модульне_тестування
5) Внедрение компонентного подхода в вебе: обзор веб-компонентов (Впровадження компонентного підходу у вебі: огляд веб-компонентів) [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/microsoft/blog/264791
6) Документація бібліотеки Vuex [Електронний ресурс] – Режим доступу: https://vuex.vuejs.org/
7) Документація фреймворку Vue [Електронний ресурс] – Режим доступу: https://vuejs.org/v2/guide/
8) Документація плагіну Vuex ORM [Електронний ресурс] – Режим доступу: https://vuex-orm.github.io/vuex-orm/
9) Документація бібліотеки Vuetify [Електронний ресурс] – Режим доступу: https://vuetifyjs.com/en/
10) Документація фреймворку NodeJS [Електронний ресурс] – Режим доступу: https://nodejs.org/en/